b441 – Fall 2006 – Test One

INSTRUCTIONS: *Put your name and ID number in the upper-right of each page. Place your final answer on the test in the space provided. Scratch work is not graded, but neatness counts.*

Question 1. Do any two of (a), (b) and (c).

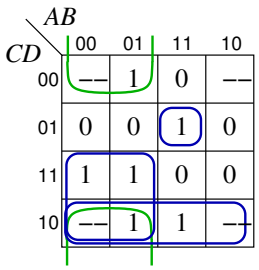
Develop minimal sum-of-products forms for the following specifications.

(a)

A	B	C	D	Z	*term*
–	0	–	0	–	don't care if $\overline{B}\ \overline{D}$
0	0	c	1	c	$\overline{A}\ \overline{B}\ D$
0	1	0	0	1	$\overline{A}\ B\ \overline{C}\ \overline{D}$
0	1	0	1	0	$\overline{A}\ B\ \overline{C}\ D$
0	1	1	0	1	$\overline{A}\ B\ C\ \overline{D}$
0	1	1	1	1	$\overline{A}\ B\ C\ D$
1	0	0	1	0	
1	0	1	1	0	
1	1	c	0	c	$A\ B\ C\ \overline{D}\ D$
1	1	c	1	$\overline{c}$	$A\ B\ \overline{C}\ D$

ANSWER: $\overline{A}\ C + \overline{A}\ \overline{D} + C\ \overline{D} + A\ B\ \overline{C}\ D$.

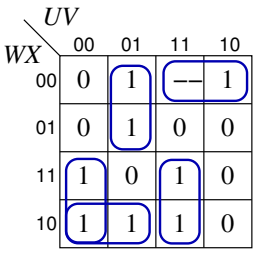
Because of the don't-cares *it is best to solve this with a K-map.*



(b)

WX \ UV	00	01	11	10
00	0	1	-	1
01	0	1	0	0
11	1	0	1	0
10	1	1	1	0

ANSWER: $\overline{U}\ V\ \overline{W} + U\ \overline{W}\ \overline{X} + \overline{U}\ \overline{W}\ X + \overline{U}\ V\ \overline{X} + U\ V\ W$



(c) $((A \cdot \overline{B}) + (C \cdot \overline{D})) \cdot (B + \overline{C} + D)$

ANSWER: $A\ \overline{B}\ \overline{C} + A\ \overline{B}\ D + C\ \overline{D}\ B$

The expression multiplies out to

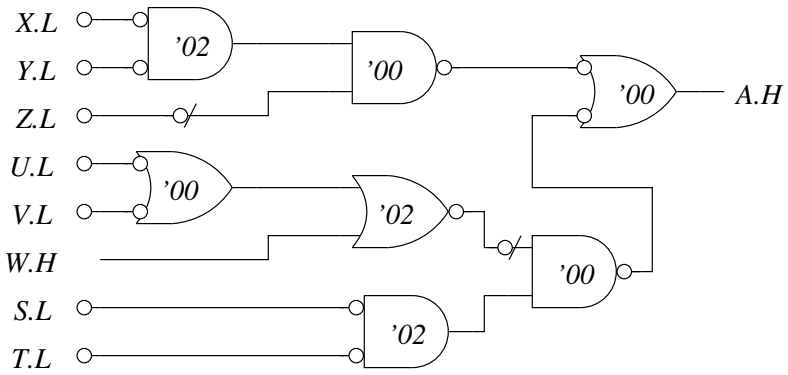
$$\begin{aligned} & ((A\ \overline{B}) + (C\ \overline{D}))\,(B + \overline{C} + D) \\ = \ & (A\ \overline{B})\,(B + \overline{C} + D) + (C\ \overline{D}))\,(B + \overline{C} + D) \\ = \ & A\,\overline{B}\,B + A\,\overline{B}\ \overline{C} + A\,\overline{B}\,D + C\,\overline{D}\,B + C\ \overline{D}\ \overline{C} + C\,\overline{D}\,D \\ = \ & A\,\overline{B}\ \overline{C} + A\,\overline{B}\,D + C\,\overline{D}\,B \end{aligned}$$

There are no further reductions of the form $xy + \overline{x}y \Rightarrow y$.

Question 2

Implement $A = ((X \cdot Y \cdot \overline{Z}) + (\overline{(U + V + W)} \cdot S \cdot T))$ *subject to the constraints that inputs X.L, Y.L, and Z.L are fixed at* true=L*; output A.H is fixed at* true=H*; and all other signals may have arbitrary polarity.*

(a) Using LS'00 (*nand*) gates, LS'02 (*nor*) gates, and LS'04 *inverters*



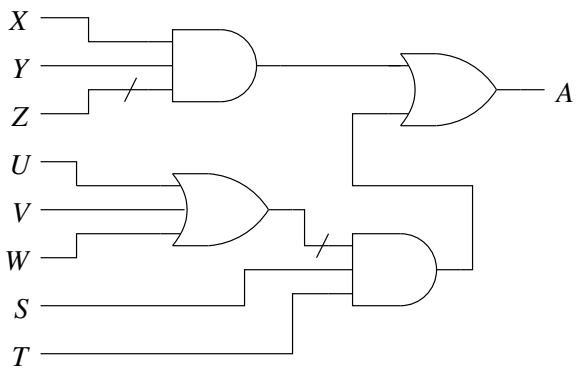
A well-formed answer should

1. *show the logic as expressed, including the* not*s Since the implementation uses 2-input logic gates,* $x + y + z$ *is interpreted as* $(x + y) + z$*.*
2. *have no inconsistent connections (polarities), including the input/output paths. Putting bubbles on the* not *symbols is optional, but helps confirm that all polarities match correctly.*
3. *correctly and consistently label the gates and inputs/outputs*

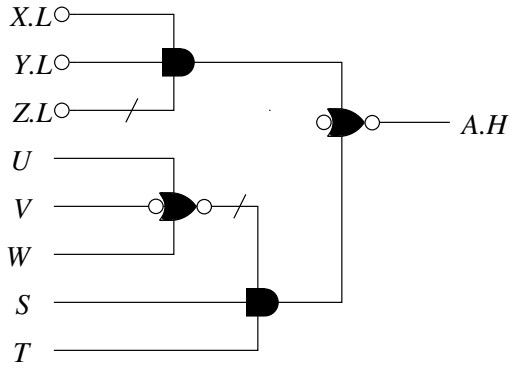
Question 2

(b) Implement $A = ((X \cdot Y \cdot \overline{Z}) + (\overline{(U + V + W)} \cdot S \cdot T))$ *using LS'06* open-collector inverters *only.*

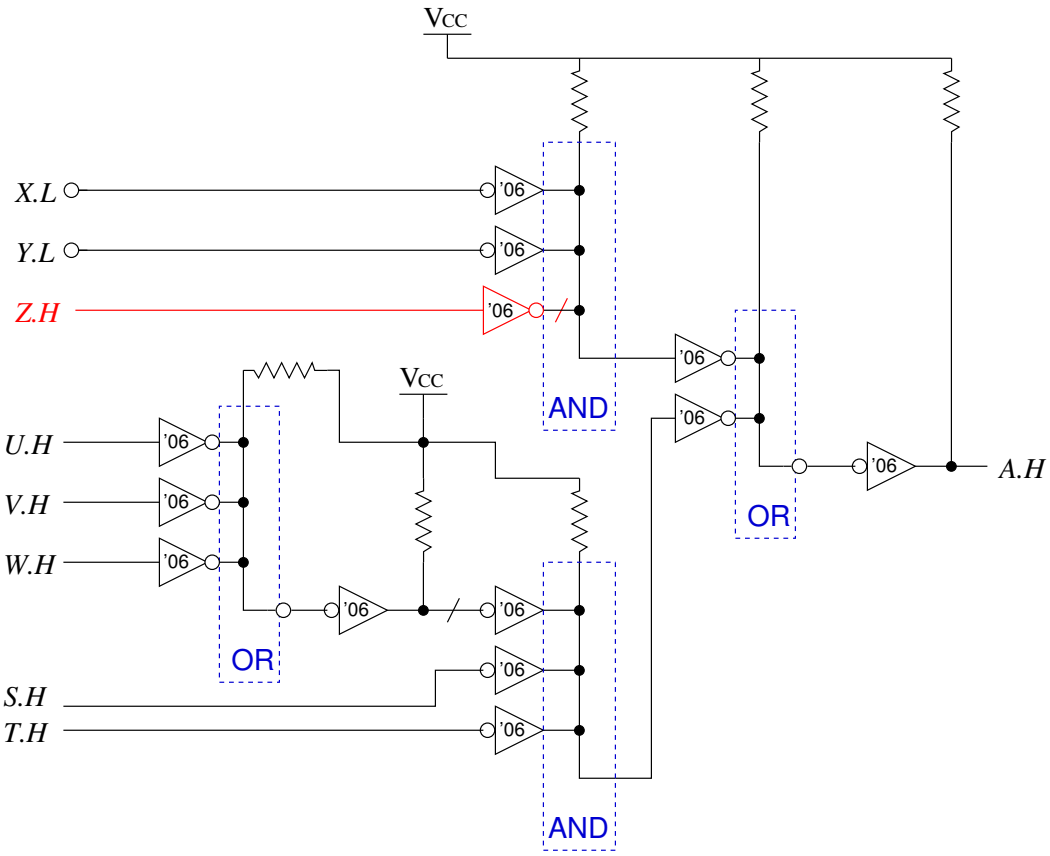
*(1) Diagram the logic as expressed. Since the implementation uses wired-*and/or *logic, gates may have arbitrarily many inputs.*



*(2) The wired-*and/or *gates provide positive-logic* and*s and negative-logic* or*s. An intermediate wired-gate diagram preserves the logic of the specification.*



*(3) Now, each wired-*and/or *must have and open-collector buffer (inverter) on* <u>every</u> *input, all tied to a pull-up resister. However, we should apply mixed logic principles, using the inverters to fix polarity mismatches. The problem as given does not illustrate the subtleties, so below I have changed the constraint on Z from* true=low *to* true=high*. By swapping the bubble on the inverter,* $\overline{Z}$ *is implemented "within" the* wired-or *gate. It doesn't matter where we put the bubble, it's the same device. Unfortunately, we can't use the same trick on the other* not *or to fix the polarity of A.*

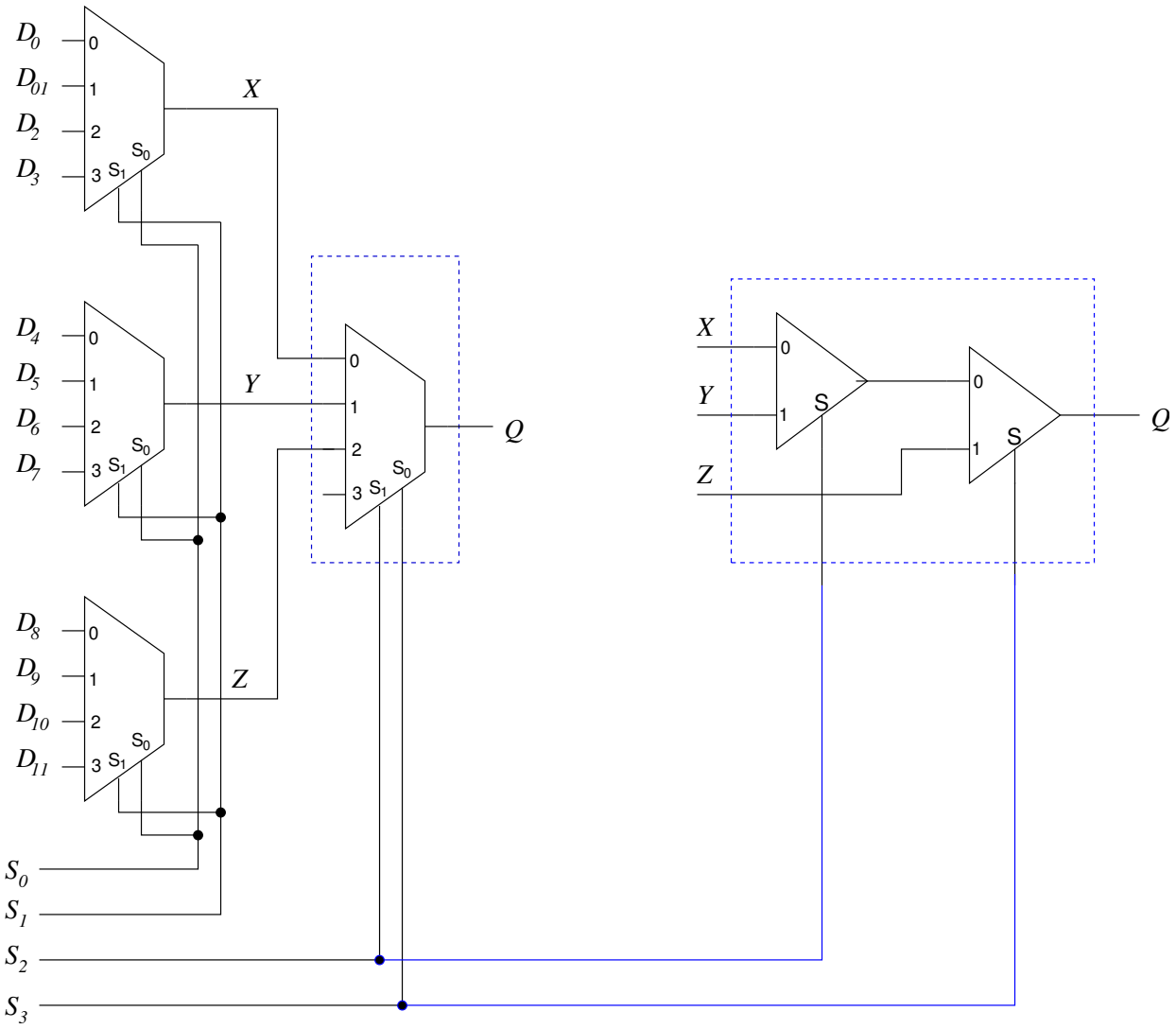


NOTE: *You are not expected to know all these details, but it is worthwhile to study and understand the application of mixed logic, as well as to remember the way open-collector connections work.*

NOTE: *It is acceptable and reasonable to solve problem using a PLA layout, which would require that you first translate the specification expression into SOP form. Of course, by doing this, you loose the logical correspondence with the original expression. I mean "loose" literally: this is the kind of design-to-implementation documentation that is often lost when someone needs to understand the design later.*

Question 3. *Design an economical* 12-input by 1-bit *multiplexor using basic combinational building blocks. Briefly explain your approach.*

One solution is to use a multiplexor tree. Depending on whether you are counting gates or chips, the 3-input selector at the second stage could be replaced by two 2-input selectors. There is an SSI package containing two 4-by-1 muxes, so you might as well use all four. Let's leave it to the surrounding system to assure that the selection word, $S_3\,S_2\,S_1\,S_0$, stays within range.



However, I was looking for a tri-state multiplexor. A decoder is used to develop ENABLE *inputs to twelve tri-state buffers whose outputs are tied to a bus. The fourth decoding bit is implemented with an* AND *gate.*

The decoding logic is roughly eqivalent in size to the mux tree above. However, these enabling signals need only be developed once, so to expand this selector to n data bits on only needs twelve additional tri-state buffers.